On Leveraging Machine and Deep Learning for Throughput Prediction in Cellular Networks: Design, Performance, and Challenges

Darijo Raca, Ahmed H. Zahran, Cormac J. Sreenan, Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, Vijay Gopalakrishnan

***Abstract*— The highly dynamic wireless communication environment poses a challenge for many applications (e.g., adaptive multimedia streaming services). Providing accurate Throughput Prediction (TP) can significantly improve performance of these applications. The scheduling algorithms in cellular networks consider various physical (PHY) metrics, (e.g., Channel Quality Indicator (CQI)), and throughput history when assigning resources for each user. This paper explains how Artificial Intelligence (AI) can be leveraged for accurate TP in cellular networks using PHY and application layer metrics. We present key architectural components and implementation options, illustrating their advantages and limitations. We also highlight key design choices and investigate their impact on prediction accuracy using real data. We believe this is the first study that examines the impact of integrating network-level data and applying a deep learning technique (on PHY and application data) for TP in cellular systems. Using video streaming as a use case, we illustrate how accurate TP improves the end user's Quality of Experience (QoE). Furthermore, we identify open questions and research challenges in the area of AI-driven TP. Finally, we report on lessons learned and provide conclusions which we believe will be useful to network practitioners seeking to apply AI.**

***Index Terms*—throughput prediction, artificial intelligence, machine learning, deep learning, HTTP adaptive video streaming, HAS, QoE**

I. Introduction

The availability of accurate TP has the potential to improve performance of many applications. For example, adaptive multimedia streaming services can improve user experience through timely quality adaptation decisions [1, 2]. Similarly, massive-scale downloads (e.g., firmware updates over the air to self-driving cars) can benefit from accurate TP by efficiently scheduling these updates without congesting the network [3]. Commonly used bandwidth estimators (e.g., exponential moving average, harmonic mean, and arithmetic mean) produce low TP accuracy, motivating the need for more advanced techniques [2].

The achievable throughput in cellular networks rapidly fluctuates due to many factors. The wireless channel features highly random characteristics. Additionally, user activity and the shared nature of the wireless medium influences the achievable throughput. Resource scheduling typically combines all these factors to trade off fair and efficient resource allocation [4] leading to throughput oscillations. Hence, the user throughput depends on various factors that are too complex to capture using traditional models (e.g., [5]). This paper focuses on using Machine and Deep Learning (ML/DL) for accurate TP in cellular networks illustrating design, performance, and challenges of this approach.

ML/DL have been very successful in tackling complex problems [6] that: (1) reflect a pattern, (2) cannot be solved mathematically or described structurally, and (3) have large amounts of example data available. The TP problem described above matches these requirements. Cellular networks provide data in abundance. Each mobile device measures and collects a wide range of control and data information, some of which are reported to the base station (BS). Additionally, cellular resource scheduling involves multitudinous parameters. While BS schedulers are vendor-specific black boxes, the scheduling algorithms represent a collection of predefined steps and actions and thus exhibit consistent behavior (i.e., for the same inputs, it will produce the same output). Hence, ML/DL represents an attractive methodology to extract underlying information and correlations in resource scheduling and make accurate TP.

II. Background

Machine and Deep Learning. ML/DL offers algorithmic methods to learn from data by extracting patterns to classify an object or predict a value. While many problems can be solved by processing raw data, extracting problem-specific features is a practical process, known as *feature engineering*. In an image recognition task, features, such as edges, corners, and ridges, are typically extracted from the image pixels. An ML algorithm then processes these features to identify the object. Defining these features for complex tasks is not trivial, and often incurs significant time and effort. To overcome this shortcoming, DL evolved as a branch of ML. A DL-based image recognition solution would take raw pixels as input and form a hierarchical structure of learning layers that extract relevant features and identify the object. Hence, DL simplifies the solution design at the cost of additional computing needs.

Throughput Prediction. Traditional TP methods rely on past throughput measurements [5]. These methods in-

Darijo Raca, Ahmed H. Zahran, and Cormac J. Sreenan are with the Department of Computer Science, University College Cork, Cork, Ireland. E-mail: {d.raca,a.zahran,cjs}@cs.ucc.ie

Darijo Raca is with the Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina. E-mail: draca@etf.unsa.ba

Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, and Vijay Gopalakrishnan are with AT&T Labs – Research, New Jersey, USA. E-mail: {sinha,emir,rjana,gvijay}@research.att.com

clude well-known linear predictors such as Moving Average (MA), Exponential Weighted Moving Average (EWMA), and more complex Autoregressive-Integrated-Moving-Average (ARIMA). The inability to detect sudden changes in throughput values is a key limitation for this type of predictors. There are also methods employing more direct expression for throughput prediction. For example, a model for estimating TCP Reno throughput uses maximum segment size, round-trip-time (RTT) and packet-loss rate. However, such model is sensitive to errors in input parameters (RTT and perceived packet loss) and usually have high prediction errors, especially in highly dynamic environments, such as cellular [5]. In contrast to traditional TP models, ML/DL can combine multidimensional input from lower and upper layers of the communication stack to provide accurate TP.

Cellular Resource Scheduling. Each cellular BS schedules transmission to its connected devices by allocating radio resources (e.g., physical resource blocks in 4G). Scheduling algorithms consider various factors in resource allocation decisions, such as reported channel quality and recently allocated resources. These schedulers are typically designed to trade-off resource fairness and efficiency [4]. Hence, combining PHY with throughput measurements can significantly improve TP accuracy. Traditional TP methods only use one metric (i.e., throughput) while the power of ML/DL is in the ability to exploit a multi-feature space for accurate TP.

III. TP System Overview

There are three main *logical* components of a TP system: the `Collector`, `Predictor` and `Trainer`, as depicted in Fig. 1. The `Collector` gathers a combination of device-level and network-level data through existing interfaces and prepares them for the `Trainer` and `Predictor`. The `Trainer` creates and updates the prediction models using training records that include both data and ground truth (actual throughput values for model training). The `Predictor` translates data records collected online to a throughput estimate (in the remainder of the text, TP refers to downlink TP).

Figure 1 also illustrates different architectural options for each of these components. These details are discussed in the following sections.

A. Collector

The `Collector` gathers various metrics that can be classified as channel-specific (e.g., CQI), network-specific (e.g., cell load), application-specific (e.g., application throughput), and context-specific (e.g., mobility mode). These metrics can also be classified as device-level and network-level data.

Device-level data represents individual device environment and can be collected at the mobile device using its Operating System (OS) Application Programming Interface (API) (e.g., Android `telephony`). Hence, the time granularity relies on OS implementation and typically has a medium time resolution (e.g., one second). It is worth noting that PHY metrics can be collected at finer granularity using specialized software or hardware equipment. Architecturally, device-based collection is distributed, and hence scalable, but only offers

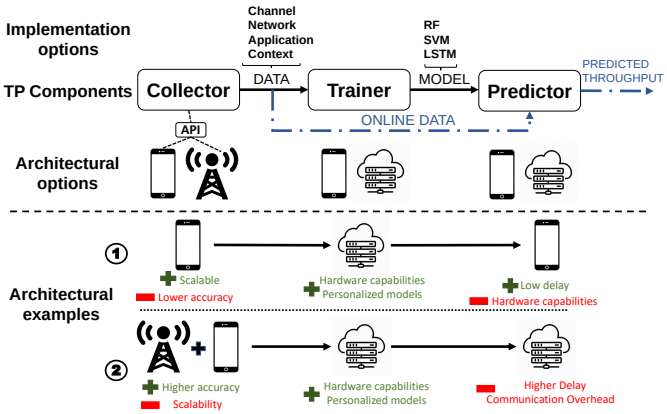

Figure 1. Components, implementation, architectural options and examples of AI-driven TP system

a local view of the operating context. Network-level data offers a comprehensive view of metrics for multiple users served by one or more BSs. Hence, TP systems that combine network-level and device-level data would have higher TP accuracy. However, lacking a unified mechanism for measuring and reporting various network-level metrics represents a key challenge in current networks (e.g., 4G). Different vendors use various approaches to collect and report relevant metrics, requiring immense effort to consolidate data from various sources.

B. Trainer & Predictor

The `Trainer` and `Predictor` are tied through the selection of the ML/DL algorithm. This design decision is affected by multiple factors, including accuracy, training time, prediction model size, and time complexity.

We select the following three algorithms as they are commonly used for TP tasks in literature [6, 2, 7, 8, 9, 10]:

- Random Forest (RF) - RF represents an ensemble learning method [7]. RF operates by constructing a set of decision trees, each trained on a random subset of training data that does not necessarily contain all the features. Hence, all decision trees see similar but different data, reducing the correlation between trees, helping avoid over-fitting. The final prediction is calculated as the mean value of all trees.
- Support Vector Machines (SVM) - SVM is similar to linear regression where output value is a linear function of input features. One of the key characteristics of SVM algorithm is the usage of kernel functions. Kernel functions enable learning non-linear interactions as a function of input features by mapping input feature space into a new transformed space. This higher-dimensional space allows fitting with the linear model [8].
- Long Short-Term Memory (LSTM) - LSTM represents a family of Recurrent Neural Networks (RNN), more specifically gated RNN. RNN is a type of DL networks design to process sequences of data and represents a suitable choice over other neural networks for time-based tasks, such as TP [6].

Typical ML algorithms heavily depend on feature engineering and model tuning. Section IV illustrates how feature engineering affects the accuracy of TP. Additionally, we compare the DL algorithm (LSTM) to ML algorithms. We find that DL models can be significantly smaller than the ML model (several Megabytes versus hundreds of Megabytes). The smaller footprint is a big advantage, especially for device-based TP systems. We perform additional measurements, implementing our DL model on an off-the-shelf smartphone (Samsung J5). The average initialization and running times of the model are 10ms and 22ms, respectively. As a result, the impact on energy consumption is negligible, making it applicable in practice. The main drawback of DL algorithms is that they require longer training times compared to ML algorithms. Therefore, training typically occurs in datacenters and it is performed only when needed.

A network-based `Trainer` can leverage the processing power of datacenters resulting in more robust prediction models. Alternatively, a device-based `Trainer` may be limited by computing and power capacities of end-devices. The `Predictor` can also run at the device or in-network. A device-based predictor would be usually limited to device-level information but facilitates low prediction delays for end-device applications. Alternatively, a network-based `Predictor` involves both delay and communication overhead but can have access to network-level data. Hence, TP systems have various architectural choices featuring different advantages and disadvantages.

Figure 1 illustrates two examples for the architecture of the TP systems. The first features a device-based `Collector` and `Predictor` with an in-network `Trainer`. The second utilizes a full network view with a network-based `Collector`, `Trainer`, and `Predictor` components.

IV. Key Design Choices and Operational Aspects

The accuracy of an ML-based TP depends on both ML-specific and system-specific design aspects. The former covers data (source, granularity, feature engineering) and algorithm choice, while the latter represents system architecture and practical implementation requirements. This section investigates the impact of key design choices on the prediction accuracy.

Our key metrics include CQI, Signal to Noise Ratio (SNR), Reference Signal Received Power (RSRP), Reference Signal Received Quality (RSRQ), device velocity and application throughput [2]. Unless mentioned otherwise, these metrics are collected using Android APIs at one second granularity. All metrics are scaled to [0,1] range as such normalization is known to improve the accuracy especially when features have different ranges.

The predictor estimates the *average* throughput over a future time horizon of x seconds by processing the data observed over the past y seconds. Therefore, notation *PyFx* is used, where Py is the Past history length and Fx is the Future prediction horizon, to illustrate the impact of history-horizon combinations on the prediction accuracy. The prediction accuracy is measured using *absolute value of relative error* (ARE). ARE is expressed as ratio of the absolute prediction error to the actual throughput value. Other relevant prediction accuracy metrics are also calculated and confirm our observations based on ARE. These include root-mean-square error (RMSE), mean absolute prediction error (MAPE), coefficient of correlation and coefficient of determination. However, they are not shared due to space limitations.

A. Feature Engineering and Machine Learning Algorithms

We show the impact of feature engineering on prediction performance by comparing two different training approaches. The first trains on *raw* data [7] while the second trains on statistical measures of raw data [2], including inter-quartile range (25^{th}, 75^{th}, 50^{th}), average and the 90^{th} percentile. The second approach is denoted as *quantile*. Note that such abstraction reduces the number of input features leading to smaller model storage requirements and shorter runtime.

Figure 2 shows the boxplot of ARE for the *raw* input and *quantile* summarization technique for different ML algorithms for P20F20.

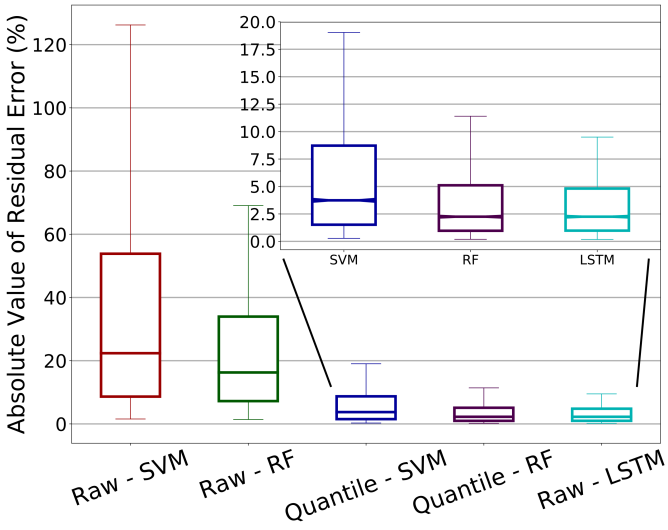

Figure 2. Comparison between different data representation approaches and ML algorithms for 1-second data granularity

The figure shows significant improvement in TP accuracy for both RF and SVM when *quantile* summarization is used. Additionally, LSTM achieves similar results operating directly on *raw* input. Furthermore, LSTM achieves the lowest 90^{th} percentile of ARE among all algorithms, indicating better learning ability for rare patterns.

B. Data granularity and history duration

There are two dimensions to how much data we use for high-level feature calculation: history duration and sampling interval. Longer history produces more samples (with fixed sampling interval) for each measurement metric, and hence improves distribution approximation. For the same prediction horizon, extending data history reduces ARE, as depicted in Fig. 3. However, in our previous work, we show that history duration beyond 20 seconds does not have a significant impact

on prediction accuracy [2]. Similarly, increasing prediction horizon improves the prediction accuracy (Fig. 3). This may appear counter-intuitive at first, but instantaneous throughput fluctuations are averaged out when the mean throughput over the horizon is calculated. Hence, increasing the horizon makes it easier for a ML algorithm to predict.

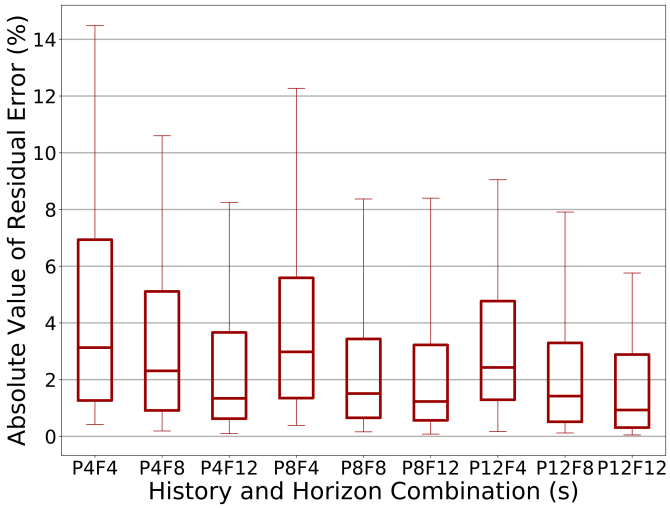

Figure 3. High granularity scenario (250ms, RF)

Finally, our results confirm that finer data granularity significantly reduces the prediction error. With 250ms data granularity (Fig. 3), which is possible using Qualcomm QXDM, ARE drops to 20-36 percent of ARE based on 1-second granularity (Device-based in Fig. 4) across all PxFy configurations.

C. Data Sources

Here we discuss the impact of integrating network-level metrics on TP accuracy. Network-level metrics are collected by monitoring logs of a group of BSs and the mobile device. The collection of these metrics involves many challenges, including scale (collecting large amounts of data), timeliness (latency), and metric reporting frequency. For example, CQI is often reported for the entire active radio session. To collect CQI, a device needs to stop using radio resources, which is achieved by stopping the device's download activity. This results in on-off measurement cycle consisting of download activity for a certain amount of time followed by the off period. We analyze TP accuracy using experiments with periodic data downloads while collecting corresponding network data. However, we did not want to make the *on* periods too long. To train and test prediction horizon of length x seconds in duration requires that we collect throughput measurements from contiguous intervals of at least x seconds. As a compromise, we used active periods of 16 seconds.

For network-level metrics, we consider following measurements:

- Competing throughput - average throughput of devices connected to a given cell
- Competing CQI, RSRP, RSRQ and SNR - average per-metric value for all devices connected to the same cell.
- Load - number of devices connected to the same cell and Physical Resource Block (PRB) utilization.

Since the number of users per cell dynamically varies with system load, we use the average value across all devices to represent competing device metrics.

Figure 4 shows significant improvement in the ARE for every PxFy combination when the TP model uses combined network-level and device-level data in comparison to using device-level data only.

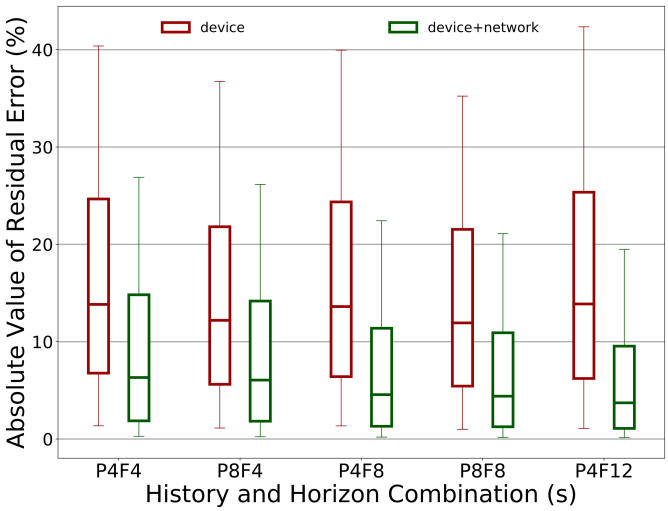

Figure 4. Comparison of ARE for device and device+network approach (aperiodic sampling interval, real data, RF)

D. History and Horizon window

The accuracy of TP is a paramount consideration in selecting a history and horizon combination, but there are additional considerations that come from application requirements and system limitation. In general, HTTP adaptive streaming service benefits more from longer horizons (dozen of seconds in future) where TP-assisted algorithms manage to stream without interruptions and with significantly fewer switches between different bitrates (qualities) than in traditional settings [2]. Longer horizons become even more important in applications needing large file download, e.g., connected cars, where size of firmware updates ranges from a couple of Megabytes to Gigabytes [3].

In any case, the choice of the horizon is driven by application/service requirements. These requirements can limit the effectiveness of the TP system if the selected horizon does not meet the desired accuracy. One approach to solving this problem is to collect more samples (either by increasing history duration or sampling interval). Furthermore, processing of large amounts of data (millions of devices) in a network may pose serious scaling challenges and increases prediction latency to a point where prediction is obsolete and unusable. However, mobile edge computing could help alleviating this challenge [6].

E. TP system in cellular networks

The implementation of device-based or network-based TP implies a different set of challenges for every design choice.

Device-based `Predictor` provides a timely and scalable option as the metrics sampling and prediction are collocated on the same device. However, loading and running large ML models on mobile devices may become impractical due to the limited memory and CPU power. The fast progress in implementing DL in mobile devices, at both software and hardware levels [6], can help overcome the model size obstacle. Alternatively, trading the model accuracy for smaller ML models could be another solution [2].

Network-based `Predictor` may be designed for device-level or combined device+network data. In both cases, the system design should consider solutions that limit the impact of communication overhead and delay, which could be critical for time-sensitive applications. Additionally, current cellular networks (4G) have limited support for deploying ML/DL at scale. The lack of a standardized way for data collection from eNodeBs across different vendors and restricted access to eNodeBs represent key challenges [11]. Next-generation cellular networks (5G) offer the opportunity for successful in-network ML/DL deployment in Mobile Edge Cloud [11] with sub-ms network-based prediction delays. Additionally, network controllers can be leveraged for data collection from Next Generation Node Base Stations (gNodeBs). Noting that network controllers govern multiple gNodeBs, they strike a balance between distributed (e.g., 4G) and centralized architecture (e.g., one controller controlling all gNodeBs) allowing 5G architecture to scale for millions of devices.

The ample computing resources available for network-based prediction can enable context-specific prediction models. To illustrate, the variation of user mobility (static to high speed) impacts throughput predictability with high mobility posing a bigger challenge for TP. By training mobility-specific models, a `Profiler` can map the device to the right model. Moreover, this classification can go beyond mobility patterns, towards personalized models. Accordingly, `Predictor` and `Trainer` would contain multiple models based on type of classification.

V. Use Case: HTTP Adaptive Video Streaming

HTTP adaptive streaming (HAS) is the dominant approach for video streaming [12]. The majority of streaming providers (e.g., YouTube, Netflix) rely on HAS for content delivery. HAS enables dynamic change of a stream's video quality during runtime by judiciously selecting the most suitable video bitrate when requesting each video chunk, allowing adjustment to changes in available bandwidth.

To illustrate the benefit of TP, we report on experiments in which a HAS video client connects to a server using a controlled link whose bandwidth is driven by a real 4G trace [2]. TP values are calculated offline using LSTM model (based on device-level data only) and are sent to the mobile device every second. In a prediction-assisted case, the video client implements ARBITER+ [13] as an adaptation algorithm. ARBITER+ uses EWMA for bandwidth estimation and scales this estimate based on the buffer level and throughput variability. For prediction-assisted streaming, we replace EWMA estimates with TP values with the rest of the algorithm's logic intact. To evaluate the video streaming performance, we calculate the following performance metrics: *Bitrate*, the average bitrate of selected video chunks; *Number of switches*, the average count of bitrate switches; *Number of stalls*, the average number of stall events, when the playback pauses due to lack of buffered chunks; *Stall duration*, the average duration of stall events. Streaming performance without prediction represents the base case. Additionally, we consider two prediction-assisted systems: the first uses the actual average throughput for 20-second horizon from the trace file (ideal prediction) while the second is based on TPs generated by P20F20 model. We repeat such evaluation ten times and report average values for each video performance metric.

Figure 5 plots the relative *improvement* in performance metrics of the prediction-assisted schemes relative to the base case. Hence, a larger relative improvement in the streaming bitrate implies a higher bitrate while a larger relative improvement in the number of stalls implies fewer stalls. The performance metrics of the base case for every scenario are shown in the white boxes just above the x-axis. Integrating TP noticeably improves QoE metrics. Ideal prediction eliminates stall events while keeping average bitrate almost as high as in the base case (insignificant 6 percent drop). Additionally, ideal prediction allows ARBITER+ to improve switching stability by having 35 percent fewer switches. Real prediction generated from ML model achieves almost the same improvement as the ideal case. Furthermore, the player operates on a distinct set of bitrates, limiting the impact of TP errors. To illustrate, inaccurate TP values have little impact as long as both ideal and real value lie in the same space between adjacent bitrates. In both cases, the player will make the same quality selection. Real prediction achieves slightly worse stall performance. Still, rebuffering events improve by 99 percent. This result reflects average bitrate as well, with real prediction achieving one percent higher bitrate than ideal case. Finally, real prediction improves switching stability by 34 percent compared to the base case. Overall, our model manages to achieve almost the same performance as ideal case with insignificant difference (one percent).

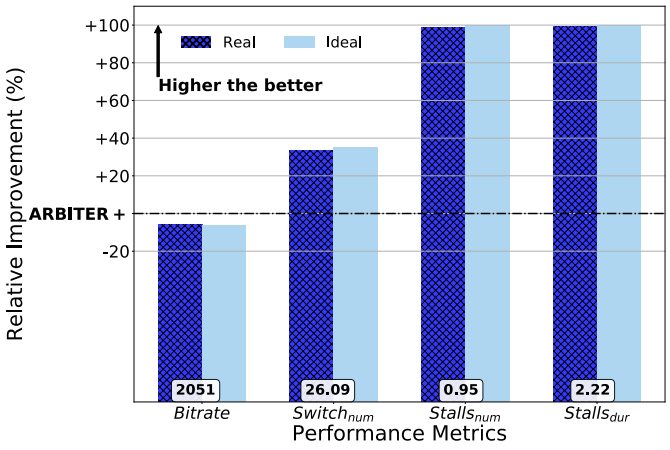

Figure 5. Relative improvement of different QoE metrics for ARBITER+ algorithm (*values are normalized to the base scenario with numbers in white boxes representing the values for the base scenario*)

Deeper insight into how TP aids the player during streaming is depicted in Fig. 6. The figure shows a timeline of one video session, comparing selected bitrates and stalls in the base and TP-assisted cases using the same bandwidth profile and video. TP-assisted player makes future-aware quality decisions in comparison to reactive decisions taken by the tradition player. As a result, the TP-assisted player eradicates stall events by swiftly switching to lower bitrate while the traditional player fails to predict sudden drops in available bandwidth and selects higher bitrates. Reducing quality switching is another benefit of TP.

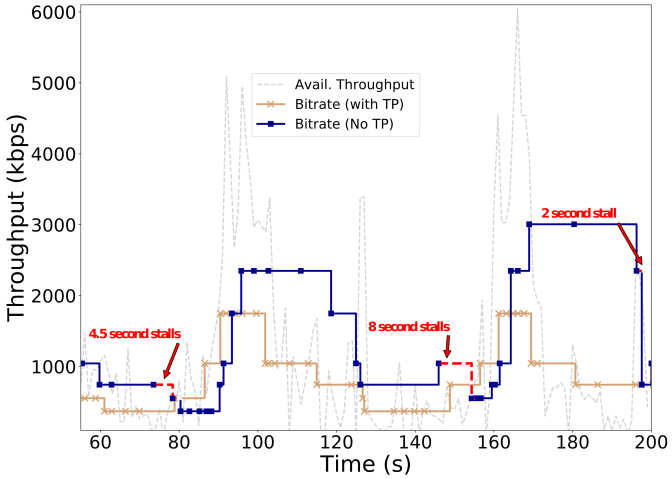

Figure 6. Timeline of video streaming sessions (with and without TP)

VI. Open Questions

Presented results indicate promising benefits from ML/DL-based TP. The results also highlight potential limitations and motivate additional research. Any ML-based TP system requires availability of metrics data, which can be challenging for various reasons. We illustrated the scalability challenge of collecting network-level data in 4G systems. Our results indicate that network data improves prediction accuracy. Hence, designing scalable network-level data collectors is an essential requirement for accurate TP. Alternatively, one can explore if the accuracy of device-based TP systems can be improved by leveraging additional PHY metrics. Also, exploring interactions between the TP system, target application and real environment is essential in order to successfully apply TP in practice.

In our study, we illustrated the impact of feature sampling granularity on the prediction accuracy. Although various metrics can be collected at different time granularities (illustrated by network-level data collection in Section III), existing studies consider a homogeneous sampling interval for various metrics. Using a summarization technique such as *quantiles* can abstract this difference in sampling interval, allowing ML models with a heterogeneous sampling of various features. However, with the deep learning approach, samples are fed directly, thus requiring more complex architectures, e.g., multi-input deep learning architectures, in order to enable TP systems with a heterogeneous sampling of features.

The implementation of ML-based solutions in real systems is known to bring new challenges that need further exploration. "Dataset shift" [14] is a common problem when training and test data come from two different distributions. This phenomenon may occur in "non-stationary environments" where training environment is different from the test one. Another possible cause is training ML model with data that does not fully represent various operational contexts (e.g., training using only mobile users that may not fully capture behavior of static users or using only a limited set of devices). The evolving learning techniques, such as transfer and active learning, can be leveraged to overcome this challenge [6].

The interplay between application traffic and metrics collection poses another challenge. Specifically, the measured throughput, which is used both as a model feature and ground truth for training, is naturally affected by the application traffic pattern. Existing studies rely on persistent traffic that saturates the link (e.g., downloading a large file) to probe the available bandwidth. However, many applications download relatively small files that may not be sufficient to probe the available system resources. An example is video streaming where the player downloads low quality segments (a few hundred Kilobytes) that may lead to lower throughput values [15]. Noting that physical-layer metrics are independent of the application, the pattern offered to the predictor may produce inaccurate predictions in such cases. Handling these situations still requires further research. One possible approach is to integrate application-specific behavior as part of the TP model.

The application using TP would benefit from knowing how much confidence to put in the prediction. For example, a video application may act conservatively if buffer occupancy is low and throughput prediction has low confidence. Identifying these scenarios also gives us motivation to develop alternative prediction techniques to handle these situations.

VII. Conclusion

ML/DL represents a promising approach for accurate TP in cellular systems. The design of ML/DL-based TP systems features a diverse set of architectural and implementation choices that require operational and performance tradeoffs. We presented a consolidated overview of these tradeoffs and their impact on the prediction accuracy. Our results indicate that suitable data representation can improve the achievable accuracy of ML algorithms with DL enhancing the learning of rare network scenarios. We also show that finer data granularity and integrating network data improve the prediction accuracy. Using video streaming as an example, we illustrate that user experience of real applications is significantly enhanced when assisted by ML/DL-based TP systems. We further highlight key open research issues that require further investigation toward the implementation of ML/DL-based TP in real networks.

Acknowledgment

This research was supported by Science Foundation Ireland (SFI) under Research Grant 13/IA/1892.

REFERENCES

[1] T. Mangla, N. Theera-Ampornpunt, M. Ammar, E. Zegura, and S. Bagchi, "Video through a crystal ball: Effect of bandwidth prediction quality on adaptive streaming in mobile environments," in *MoVid '16*. ACM, 2016, pp. 1:1–1:6.

[2] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, V. Gopalakrishnan, B. Bathula, and M. Varvello, "Empowering video players in cellular: Throughput prediction from radio network measurements," in *MMSys '19*. ACM, 2019, pp. 201–212.

[3] C. E. Andrade, S. D. Byers, V. Gopalakrishnan, E. Halepovic, M. Majmundar, D. J. Poole, L. K. Tran, and C. T. Volinsky, "Managing massive firmware-over-the-air updates for connected cars in cellular networks," in *CarSys '17*. ACM, 2017, pp. 65–72.

[4] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in lte cellular networks: Key design issues and a survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 678–700, 2013.

[5] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer tcp throughput," in *SIGCOMM '05*. ACM, 2005, pp. 145–156.

[6] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[7] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, and W. Wei, "Linkforecast: Cellular link bandwidth prediction in lte networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1582–1594, 2018.

[8] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction," *IEEE/ACM Transactions on Networking*, vol. 18, no. 4, pp. 1026–1039, 2010.

[9] L. Mei, R. Hu, H. Cao, Y. Liu, Z. Han, F. Li, and J. Li, "Realtime mobile bandwidth prediction using lstm neural network," in *PAM'19*. Springer, 2019, pp. 34–47.

[10] A. Samba, Y. Busnel, A. Blanc, P. Dooze, and G. Simon, "Predicting file downloading time in cellular network: Large-scale analysis of machine learning approaches," *Computer Networks*, vol. 145, pp. 243–254, 2018.

[11] M. Polese, R. Jana, V. Kounev, K. Zhang, S. Deb, and M. Zorzi, "Machine Learning at the Edge: A Data-Driven Architecture with Applications to 5G Cellular Networks," *arXiv e-prints*, p. arXiv:1808.07647, Aug 2018.

[12] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over http," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 562–585, 2019.

[13] A. H. Zahran, D. Raca, and C. J. Sreenan, "Arbiter+: Adaptive rate-based intelligent http streaming algorithm for mobile networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2716–2728, 2018.

[14] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition*, vol. 45, no. 1, pp. 521 – 530, 2012.

[15] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of lte: Effect of network protocol and application behavior on performance," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 363–374, 2013.

Darijo Raca is a teaching assistant at University of Sarajevo. He received his Ph.D. from University College Cork in 2020. His current research interests include multimedia networking, wireless and cellular systems, and data-driven network designs.

Ahmed H. Zahran is a lecturer at the Department of Computer Science, University College Cork. He received his Ph.D. from the University of Toronto in 2007. His current research focuses on the design and optimization of wireless networks and applications.

Cormac J. Sreenan is a professor of computer science at University College Cork in Ireland, where he also serves as head of school. Previously he was on the research staff at AT&T (Bell) Labs. He holds a Ph.D. in computer science from the University of Cambridge.

Rakesh K. Sinha is a Lead Inventive Scientist at AT&T Labs–Research. He received his B.Tech. from Indian Institute of Technology (I.I.T.), Kanpur, India and his Ph.D. from University of Washington, Seattle. He has broad research interests in the areas of network architecture, design, and optimization.

Emir Halepovic is a Principal Inventive Scientist at AT&T Labs–Research. He received his Ph.D. from the University of Calgary, Canada. His interests are in the areas of networking, wireless, content delivery, video streaming, cross-layer interactions, quality of experience and data analytics.

Rittwik Jana is a Director of Inventive Science at AT&T Labs Research. His research interests span architecting the disaggregated RAN intelligent controller in O-RAN, video streaming and cellular networks and systems. Rittwik earned a Ph.D. in Telecommunications Engineering from the Australian National University, Canberra, Australia in 2000.

Vijay Gopalakrishnan is a Director at AT&T Labs—Research. He received his Ph.D. in computer science from the University of Maryland, College Park, MD, in 2006. Vijay leads a team of researchers focused on systems aspects of networking, network management, and content delivery.